



BMC Management

Table of contents

CEC and BMC Firmware Operations	3
Boot Sequence Overview	35
User Management	37
LLDP in Redfish	47
BMC Monitoring	50
BMC FRUs	50
Product Instance Identifier	52
Software Versioning	52
Storage Health Monitoring	54
BMC Reset Control	61
Factory Reset BMC	61
Reset or Reboot BMC	63
BMC Networking	64
Guest Tunnel	64
Network Protocol Support	67
BMC Redfish	79
BlueField BMC Redfish Triggers	79
Redfish Certificate Management	80

NVIDIA BMC is based on the OpenBMC open-software framework which builds a complete Linux image for a board management controller (BMC). It uses the Yocto project as the underlying building and distro generation framework.

The primary software components of BMC are the following:

- U-boot bootloader
- Linux kernel
- OpenBMC distro

This section consists of the following pages:

- [CEC and BMC Firmware Operations](#)
- [Boot Sequence Overview](#)
- [User Management](#)
- [LLDP in Redfish](#)
- [BMC Monitoring](#)
- [BMC Reset Control](#)
- [BMC Networking](#)
- [BMC Redfish](#)

CEC and BMC Firmware Operations

Firmware upgrade of BMC and CEC components using BMC can be performed from a remote server using the Redfish interface.



CEC and BMC Firmware Commands

Triggering Secure Firmware Update

i Info

Required for BMC/CEC update.

This section describes how to trigger a secure update and to track the secure update progress.

- To perform a multipart update with `MultipartHttpPushUri` API, run:

```
curl -k -u root:'<password>'
https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters={};type=application/octet-stream' -F
UpdateFile=@<package_path>
```

- Update with `HttpPushUri` API – Deprecated

 **Warning**


`HttpPushUri` API is obsolete. NVIDIA recommends users migrate to `MultipartHttpPushUri` API as support for `HttpPushUri` API will be discontinued in the future.

```
curl -k -u root:'<password>' -H "Content-Type:
application/octet-stream" -X POST -T <package_path>
https://<bmc_ip>/redfish/v1/UpdateService/update
```

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address
- `package_path` – firmware update package path

Tracking Secure Firmware Update Progress

 **Info**

Required for BMC/CEC update.

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/TaskService/Tasks
```

Find the current task ID in the response and use it for checking the progress by running:

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/TaskService/Tasks/<task_id> | jq -r '
.PercentComplete'
```

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address
- `task_id` – Task ID

To retrieve additional information about the task progress, run:

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/TaskService/Tasks/<task_id>
```

The task status can be one of the following:

- `"Running"` – the update is currently in progress.
- `"Completed"` – the update finished successfully.
- `"Exception"` – an error occurred during the update.

Resetting/Rebooting BMC

Info

Required for BMC update.

To reset/reboot BMC, run:

```
curl -k -u root:'<password>' -H "Content-Type: application/json"
-X POST -d '{"ResetType": "GracefulRestart"}'
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Actions/Manager
```

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address

Getting the Running BMC Firmware Version

Info

Required for BMC update.

The following commands get the running firmware version of BlueField devices via the BMC.

- For NVIDIA® BlueField®-3:

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/BM
| jq -r ' .Version'
```

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address
- For NVIDIA® BlueField®-2:

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory
```

Get the current firmware ID and then perform:

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/<f
| jq -r ' .Version'
```

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address
- `firmware_id` – numeric value found in the `FwInventory` schema only. It is calculated during firmware update by the BMC and used to distinguish between the versions.

Getting the Running CEC Firmware Version

Info

Required for CEC update.

Gets the running firmware version from CEC.

```
curl -k -u root:'<password>' -X GET  
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/Bluefield  
| jq -r '.Version'
```

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address

ForceUpdate

Info

Relevant for BlueField-3 only.

i Info

Required for BMC update.

`ForceUpdate` forces the update procedure to proceed even if the target version is identical to the currently programmed version.

- When using `HttpPushUri` API, `ForceUpdate` is always `true` (i.e., forces the update procedure to proceed even if the target version is identical)
- When using `MultipartHttpPushUri` API, `ForceUpdate` is `false` (i.e., the update procedure fails if the target version is identical), unless `"ForceUpdate":true` is included under `UpdateParameters`:

```
curl -k -u root:'<password>'
https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters=
{"ForceUpdate":true};type=application/octet-stream' -F
UpdateFile=@<package_path>
```

Where:

- - `password` – password of root user
 - `bmc_ip` – BMC IP address

Updating BMC

Info

Firmware update takes about 12 minutes.

After initiating the BMC secure update, you can expect to receive responses similar to the following.

- If an `HttpPushUri` API is used:

```
curl -k -u root:'<password>' -H "Content-Type: application/octet-stream" -X POST -T <package_path>
https://<bmc_ip>/redfish/v1/UpdateService

{
  "@odata.id": "/redfish/v1/TaskService/Tasks/0",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "<id>",
  "TaskState": "Running"
}
```

- If a `MultipartHttpPushUri` API is used:

```
curl -k -u root:'<password>'
https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters={};type=application/octet-stream' -F
UpdateFile=@<package_path>
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/0",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "<id>",
  "TaskState": "Running",
  "TaskStatus": "OK"
}
```

Where:

- - `package_path` – the BMC firmware image file including its path

For example:

```
curl -k -u root:'myP@ssword_12345!'
https://10.10.10.10/redfish/v1/UpdateService/update-multipart -F 'UpdateParameters=
{};type=application/octet-stream' -F UpdateFile=@/root/bf2-bmc-ota-24.01-4-ipn.tar
```

The following command is used to track secure firmware update progress:

```

curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/TaskService/Tasks/<id> | jq -r '
.PercentComplete'

  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current Dload  Upload    Total   Spent    Left   Speed
100 2123 100 2123    0     0 38600      0 --:--:-- --:--:-- -
-:--:-- 37910
20

```

The task is completed when `PercentComplete` reaches 100.

Since the reboot option is disabled during the update procedure, use the following command to reboot the BMC.

```

curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/TaskService/Tasks/<id> | jq -r '
.PercentComplete'
  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current Dload  Upload    Total   Spent    Left   Speed
100 3822 100 3822    0     0 81319      0 --:--:-- --:--:-- -
-:--:-- 81319
100

curl -k -u root:'<password>' -H "Content-Type: application/octet-
stream" -X POST -d '{"ResetType": "GracefulRestart"}'
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Actions/Manager
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The request completed successfully.",
      "MessageArgs": [],
      "MessageId": "Base.1.13.0.Success",
      "MessageSeverity": "OK",
      "Resolution": "None"
    }
  ]
}

```

The following commands are used to verify the current BMC firmware version after reboot:

- For BlueField-3:

```

curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/BMC
| jq -r '.Version'

% Total    % Received % Xferd  Average Speed   Time    Time
Time  Current Dload  Upload    Total   Spent    Left   Speed
100  513  100  513    0     0  9679      0 --:--:-- --:--
-:-- --:--:--  9679

```

- For BlueField-2:

1. Get the firmware ID from `FirmwareInventory`.

```

curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory
{
  "@odata.id" :
  "/redfish/v1/UpdateService/FirmwareInventory",
  "@odata.type" :
  "#SoftwareInventoryCollection.SoftwareInventoryCollection"
  "Members": [
    {
      "@odata.id" :
      "/redfish/v1/UpdateService/FirmwareInventory/8c8549f3_BMC_
...

```

2. Use the following command with the retrieved firmware ID from the previous step.

```

curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory
| jq -r ' .Version'

      % Total    % Received % Xferd  Average Speed   Time
Time      Time     Current
           Dload  Upload   Total
Spent     Left  Speed
100    471    100    471    0     0    622      0  --:--:--  -
-:--:--  -:--:--    621
bmc-23.04

```

For BlueField-3 BMC only: When updating firmware to a target that has an identical version using `MultipartHttpPushUri`, you must include `ForceUpdate=true`. Otherwise, the update procedure fails with the message `Component image is identical`.

```

curl -k -u root:'<password>'
https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters={"ForceUpdate":true};type=application/octet-
stream' -F UpdateFile=@<package_path>

```

Updating CEC

Info

Firmware update takes about 20 seconds.

After initiating the BMC secure update, expect responses similar to the following-- depending whether `HttpPushUri` or `MultipartHttpPushUri` is used:

- `HttpPushUri` API:

```
curl -k -u root:'<password>' -H "Content-Type:
application/octet-stream" -X POST -T <package_path>
https://<bmc_ip>/redfish/v1/UpdateService
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/0",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "0",
  "TaskState": "Running"
```

- `MultipartHttpPushUri` API:

```
curl -k -u root:'<password>'
https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters={};type=application/octet-stream' -F
UpdateFile=@<package_path>
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/0",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "0",
  "TaskState": "Running",
  "TaskStatus": "OK"
}
```

Where:

- `package_path` – the BMC firmware image file including its path

For example:

```
curl -k -u root:'myP@ssword_12345!'
https://10.10.10.10/redfish/v1/UpdateService/update-
multipart -F 'UpdateParameters=
{};type=application/octet-stream' -F
UpdateFile=@/root/cec_ota_BMGP-04.0f_debug.bin
```

Use the following command to track the progress of the CEC firmware update.

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/TaskService/Tasks/0 | jq -r '
.PercentComplete'
  % Total      % Received % Xferd  Average Speed   Time    Time
Time  Current Dload  Upload    Total   Spent    Left   Speed
100  2123  100  2123    0     0  38600      0 --:--:-- --:--:-- -
-:--:-- 37910
100
```

Note

After the CEC secure update operation is complete, CEC activation and reset should be done (see "[Activating New CEC](#)") to apply the changes once the update is finished.

Use the following command to verify the current CEC firmware version after reboot.

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/Bluefield
| jq -r '.Version'
```

```

% Total    % Received % Xferd  Average Speed   Time    Time
Time  Current
                                Dload  Upload  Total  Spent
Left  Speed
100  421  100  421    0    0   1172    0 --:--:-- --:--:--
-:--:-- 1172
19-4
```

Activating New CEC

Note

This is relevant only for BlueField-3 networking platforms (DPU or SuperNIC).

To activate the new CEC firmware, you must reset the CEC device. The following subsections describe possible reset options.

Resetting CEC and BMC Subsystems Using CEC Self-reset Command

Note

The CEC self-reset command is supported in CEC versions `00.02.0180.0000` and above. The command activates the new firmware on CEC and should only be used after the CEC update procedure is complete.

Redfish Command

```
curl -k -u root:'<password>' -H "Content-Type: application/json"
-X POST -d '{"ResetType": "GracefulRestart"}'
https://<bmc_ip>/redfish/v1/Chassis/Bluefield_ERoT/Actions/Chassis
```

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address

Command response options:

- The response in case of success:

```
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The request completed successfully.",
      "MessageArgs": [],
      "MessageId": "Base.1.15.0.Success",
      "MessageSeverity": "OK",
      "Resolution": "None"
    }
  ]
}
```

- The response if there is no pending CEC firmware:

```

curl -k -u root:'<password>' -H "Content-Type:
application/json" -X POST
https://<bmc_ip>/redfish/v1/Chassis/Bluefield_ERoT/Actions/Cha
-d '{"ResetType":"GracefulRestart"}'
{
  "error": {
    "@Message.ExtendedInfo": [
      {
        "@odata.type": "#Message.v1_1_1.Message",
        "Message": "The requested resource of type ERoT FW
named 'Pending-ERoT-FW' was not found.",
        "MessageArgs": [
          "ERoT FW",
          "Pending-ERoT-FW"
        ],
        "MessageId": "Base.1.15.0.ResourceNotFound",
        "MessageSeverity": "Critical",
        "Resolution": "Provide a valid resource identifier
and resubmit the request."
      }
    ],
    "code": "Base.1.15.0.ResourceNotFound",
    "message": "The requested resource of type ERoT FW named
'Pending-ERoT-FW' was not found."
  }
}

```

- The response if the command is not supported by the current CEC firmware:

```
curl -k -u root:'<password>' -H "Content-Type:
application/json" -X POST
https://<bmc_ip>/redfish/v1/Chassis/Bluefield_ERoT/Actions/Cha
-d '{"ResetType":"GracefulRestart"}'
{
  "error": {
    "@Message.ExtendedInfo": [
      {
        "@odata.type": "#Message.v1_1_1.Message",
        "Message": "The action ERoT self-reset is not
supported by the resource.",
        "MessageArgs": [
          "ERoT self-reset"
        ],
        "MessageId": "Base.1.15.0.ActionNotSupported",
        "MessageSeverity": "Critical",
        "Resolution": "The action supplied cannot be
resubmitted to the implementation. Perhaps the action was
invalid, the wrong resource was the target or the
implementation documentation may be of assistance."
      }
    ],
    "code": "Base.1.15.0.ActionNotSupported",
    "message": "The action ERoT self-reset is not supported
by the resource."
  }
}
```

IPMI Command

```
ipmitool raw 0x32 0xD2
```

Command response options:

- If successful, no response is given. Glacier and BMC are reset.
- The response if there is no pending CEC firmware is:

```
Unable to send RAW command (channel=0x0 netfn=0x32 lun=0x0  
cmd=0xd2 rsp=0xd6): Cannot execute command, command disabled
```

- The response if the command is not supported by the current CEC firmware is:

```
Unable to send RAW command (channel=0x0 netfn=0x32 lun=0x0  
cmd=0xd2 rsp=0xd5): Command not supported in present state
```

Log Event Entries Created per Response Type

- Log event entry created in case of success:

```

{
  "@odata.id":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "Created": "<Date>",
  "EntryType": "Event",
  "Id": "<Id>",
  "Message": "The request completed successfully.",
  "MessageArgs": [
    ""
  ],
  "MessageId": "Base.1.0.Success",
  "Name": "System Event Log Entry",
  "Resolution": "Ready for ERoT self Reset",
  "Resolved": false,
  "Severity": "OK"
}

```

- Log event entry created if there is no pending CEC firmware:

```

curl -k -u root:'<password>' -H "Content-Type:
application/json" -X GET
https://<bmc_ip>/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>
{
  "@odata.id":
"/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI":
"/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "Created": "<Date>",
  "EntryType": "Event",
  "Id": "<Id>",
  "Message": "Awaiting for an action to proceed with
installing image '' on 'ERoT'.",
  "MessageArgs": [
    "",
    "ERoT"
  ],
  "MessageId": "Update.1.0.AwaitToUpdate",
  "Name": "System Event Log Entry",
  "Resolution": "Cannot perform ERoT self reset: There is no
EC FW pending. Perform an ERoT FW update.",
  "Resolved": false,
  "Severity": "OK"
}

```

- Log event entry created if the command is not supported by the current CEC firmware:

```

{
  "@odata.id":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "Created": "<Date>",
  "EntryType": "Event",
  "Id": "<Id>",
  "Message": "The action ERoT self Reset is not supported by
the resource.",
  "MessageArgs": [
    "ERoT self Reset"
  ],
  "MessageId": "Base.1.0.ActionNotSupported",
  "Name": "System Event Log Entry",
  "Resolution": "Cannot perform ERoT self reset: The action
is not supported by the current ERoT version.",
  "Resolved": false,
  "Severity": "OK"
}

```

- Possible log event entries created if an error occurs during the BMC shutdown procedure (received after a success log event entry):

```
{
  "@odata.id":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<I
  ...
  "MessageId": "Base.1.0.InternalError",
  "Name": "System Event Log Entry",
  "Resolution": "ERoT Reset: Failed to close services towards
ERoT reset",
  "Resolved": false,
  "Severity": "Critical"
}
```

```
{
  "@odata.id":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<I
  ...
  "MessageId": "Base.1.0.InternalError",
  "Name": "System Event Log Entry",
  "Resolution": "ERoT Reset: Isolate operation may still be
in progress.",
  "Resolved": false,
  "Severity": "Critical"
}
```

```
{
  "@odata.id":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<I
  ...
  "MessageId": "Base.1.0.InternalError",
  "Name": "System Event Log Entry",
  "Resolution": "ERoT Reset: Failed to unmount file system
  towards ERoT reset.",
  "Resolved": false,
  "Severity": "Critical"
}
```

```
{
  "@odata.id":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<I
  ...
  "MessageId": "Base.1.0.InternalError",
  "Name": "System Event Log Entry",
  "Resolution": "ERoT Reset: Failed to unmount file system. It
  is still mounted as read-write (rw),
  "Resolved": false,
  "Severity": "Critical"
}
```

Resetting CEC and BMC Subsystems Using IPMI I2C Command over SMBus Channel Connected to PCIe Golden Finger

i Note

This option is valid only for servers which support I²C over SMBus from the host BMC.

```
ipmitool raw 0x06 0x52 <BUS-ID> 0x82 0x00 0x03 0xFE
ipmitool raw 0x06 0x52 <BUS-ID> 0x82 0x00 0x01 0xFE
sleep <100ms>
ipmitool raw 0x06 0x52 <BUS-ID> 0x82 0x00 0x01 0xFF
ipmitool raw 0x06 0x52 <BUS-ID> 0x82 0x00 0x03 0xFF
```

i Info

The `BUS-ID` value is system related. It relays how the host BMC is connected to the SMBus of the related BlueField.

i Info

The format of the `ipmitool i2c` command is as follows:

```
ipmitool raw <netfun> <cmd> <bus-id> <addr>
<read-count> <write-data1> <write-data2>
```

Resetting the Entire BlueField

This option typically involves a full power cycle of the host platform.

CEC Background Update Status

Info

This section is relevant only for BlueField-3.

BMC and CEC have an active and inactive copy of the same firmware image on their respective firmware SPI flash devices. The firmware update is performed on the inactive copy. Upon a successful boot from the newly updated and active image, the inactive image (e.g., the previously active image) is updated with the latest image.

Note

Firmware update cannot be initiated if the background copy is in progress.

To check the status of the background update, run:

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/Chassis/Bluefield_ERoT
...
  "Oem": {
    "Nvidia": {
      "@odata.type": "#NvidiaChassis.v1_0_0.NvidiaChassis",
      "AutomaticBackgroundCopyEnabled": true,
      "BackgroundCopyStatus": "Completed",
      "InbandUpdatePolicyEnabled": true
    }
  }
...

```

Info

The background update initially indicates `InProgress` while the inactive copy of the image is being updated.

Possible Error Codes

Info

This section is relevant only for BlueField-3.

Fault	Diagnosis and Possible Solution
<p>Connection to BMC breaks during firmware package transfer</p>	<ul style="list-style-type: none"> • Redfish task URI is not returned by the Redfish server • The Redfish server (if operational) is in <code>idle</code> state • After a reboot of BMC, or after restart/recovery of the Redfish server, the Redfish server is in <code>idle</code> state <p>A new firmware update can be attempted by the Redfish client.</p>
<p>Connection to BMC breaks during firmware update</p>	<ul style="list-style-type: none"> • Redfish task URI that was previously returned by the Redfish server is no longer accessible • The Redfish server (if operational) is in one of the following states: <ul style="list-style-type: none"> ◦ In <code>idle</code> state, if the firmware update has completed ◦ In <code>update</code> state, if the firmware update is still ongoing • After a BMC reboot, or after restart/recovery of the Redfish server, the Redfish server is in <code>idle</code> state <p>A new firmware update can be attempted by the Redfish client.</p>
<p>Two firmware update requests are initiated</p>	<p>The Redfish server blocks the second firmware update request and returns the following:</p> <ul style="list-style-type: none"> • HTTP code 400 "Bad Request" • Redfish message based on standard registry entry <code>UpdateInProgress</code> • A resolution is proposed: "Another update is in progress. Retry the update operation once it is complete." <p>Check the status of the ongoing firmware update by looking at the <code>TaskCollection</code> resource.</p>
<p>Redfish task hangs</p>	<ul style="list-style-type: none"> • Redfish task URI that was previously returned by the Redfish server is no longer accessible • PLDM-based firmware update progresses • After a reboot of BMC, or after restart/recovery of the Redfish server, the Redfish server is in <code>idle</code> state <p>A new firmware update can be attempted by the Redfish client.</p>

Fault	Diagnosis and Possible Solution
BMC-EROT communication failure during image transfer	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • <code>TaskState</code> is set to <code>Exception</code> • <code>TaskStatus</code> is set to <code>Warning</code> • <code>Messages</code> array in the task includes an entry based on the standard registry <code>Update.1.0.0.TransferFailed</code> indicating the components that failed during image transfer <p>The Redfish client may retry the firmware update.</p>
Firmware update fails	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • <code>TaskState</code> is set to <code>Exception</code> • <code>TaskStatus</code> is set to <code>Warning</code> • <code>Messages</code> array in the task includes an entry describing the error <p>The Redfish client may retry the firmware update.</p>
ERoT failure (not responding)	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • <code>TaskState</code> is set to <code>Canceled</code> • <code>TaskStatus</code> is set to <code>Warning</code> • <code>Messages</code> array in the task includes an entry describing the error • The Redfish client reports the error <p>The Redfish client may retry the firmware update.</p>

Fault	Diagnosis and Possible Solution
Firmware image validation failure	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • <code>TaskState</code> is set to <code>Exception</code> • <code>TaskStatus</code> is set to <code>Warning</code> • <code>Messages</code> array in the task includes an entry based on the standard registry <code>Update.1.0.0.VerificationFailed</code> to indicate the component for which verification failed • The Redfish client reports the error <p>The Redfish client might retry the firmware update.</p>
Power loss before activation command is sent	<ul style="list-style-type: none"> • The Redfish server is in <code>idle</code> state <p>A new firmware update can be attempted by the Redfish client.</p>
Firmware activation failure	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • <code>TaskState</code> is set to <code>Exception</code> • <code>TaskStatus</code> is set to <code>Warning</code> • <code>Messages</code> array in the task includes an entry based on the standard registry <code>Update.1.0.ActivateFailed</code> <p>The Redfish client may retry the firmware update.</p>
Push to BMC firmware package greater than 200 MB	<ul style="list-style-type: none"> • No Redfish task is created • <code>Messages</code> array in the task includes an entry based on the standard registry <code>Base.1.15.0.PayloadTooLarge</code> and the <code>Resolution</code> "Firmware package size is greater than allowed size". Make sure the package size is less than the <code>UpdateService.MaxImageSizeBytes</code> property and retry the firmware update operation.

Boot Sequence Overview

1. BMC starts booting through u-boot bootloader once the power supply is powered on.
2. By default, the BMC automatically boots into Linux. To stop at the u-boot prompt, users must type the password `0penBmc` (note the use of the digit zero in `0pen`) within 5 seconds. To boot Linux from the u-boot prompt, type `boot`.
3. The BMC provides indications of its status during its operation:

Scenario	Message
At the beginning of the boot process of the u-boot	<pre>Nvidia Bluefield BMC U-BOOT starting</pre>
At the beginning of the OS boot process	<pre>Nvidia Bluefield BMC Starting kernel ...</pre>
At the login prompt	<pre>Nvidia Bluefield BMC OS is up and running</pre>
Upon reboot or shutdown	<pre>Nvidia Bluefield BMC is shutting down</pre>

4. The default password for the root user, to be typed in once Linux is booted, is `0penBmc`.

Info

For information on password policy, refer to section "[BMC Management Interface](#)".

User Management

User Management Redfish Commands

Getting General Information

To retrieve general information about the BMC account services:

```
curl -k -u root:'<password>' -H 'Content-Type: application/json'  
-X GET https://<IP>/redfish/v1/AccountService
```

Example output:

```

{
  "@odata.id": "/redfish/v1/AccountService",
  "@odata.type": "#AccountService.v1_10_0.AccountService",
  "AccountLockoutDuration": 600,
  "AccountLockoutThreshold": 4,
  "Accounts": {
    "@odata.id": "/redfish/v1/AccountService/Accounts"
  },
  ..
  "MaxPasswordLength": 20,
  "MinPasswordLength": 13,
  "Name": "Account Service",
  "Oem": {
  ..
  "Roles": {
    "@odata.id": "/redfish/v1/AccountService/Roles"
  },
  "ServiceEnabled": true
}

```

Listing Supported User Roles

To list supported user roles in the system:

```

curl -k -u root:'<password>' -H 'Content-Type: application/json'
-X GET https://<IP>/redfish/v1/AccountService/Roles

```

Example output:

```

{
  "@odata.id": "/redfish/v1/AccountService/Roles",
  "@odata.type": "#RoleCollection.RoleCollection",
  "Description": "BMC User Roles",
  "Members": [
    {
      "@odata.id":
"/redfish/v1/AccountService/Roles/Administrator"
    },
    {
      "@odata.id": "/redfish/v1/AccountService/Roles/Operator"
    },
    {
      "@odata.id": "/redfish/v1/AccountService/Roles/ReadOnly"
    },
    {
      "@odata.id": "/redfish/v1/AccountService/Roles/NoAccess"
    }
  ],
  "Members@odata.count": 4,
  "Name": "Roles Collection"
}

```

Listing User Accounts

```

curl -k -u root:'<password>' -H 'Content-Type: application/json'
-X GET https://<IP>/redfish/v1/AccountService/Accounts

```

Example output:

```

{
  "@odata.id": "/redfish/v1/AccountService/Accounts",
  "@odata.type":
"#ManagerAccountCollection.ManagerAccountCollection",
  "Description": "BMC User Accounts",
  "Members": [
    {
      "@odata.id":
"/redfish/v1/AccountService/Accounts/NvdBluefieldUefi",
    },
    {
      "@odata.id": "/redfish/v1/AccountService/Accounts/root"
    }
  ],
  "Members@odata.count": 2,
  "Name": "Accounts Collection"
}

```

Creating New User

To create a new user on the BMC:

```

curl -k -u root:'<password>' -H 'Content-Type: application/json'
-X POST https://<IP>/redfish/v1/AccountService/Accounts -d '{
"UserName": "<USER>", "Password": "<PASSWORD>", "RoleId": "<ROLE>",
"Enabled": true}'

```

Example output:

```
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The resource has been created successfully.",
      "MessageArgs": [],
      "MessageId": "Base.1.15.0.Created",
      "MessageSeverity": "OK",
      "Resolution": "None."
    }
  ]
}
```

Deleting User

To delete a user from the system:

```
curl -k -u root:'<password>' -H 'Content-Type: application/json'
-X DELETE https://<IP>/redfish/v1/AccountService/Accounts/<USER>
```

Example:

```
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The account was successfully removed.",
      "MessageArgs": [],
      "MessageId": "Base.1.15.0.AccountRemoved",
      "MessageSeverity": "OK",
      "Resolution": "No resolution is required."
    }
  ]
}
```

User Management IPMI Commands

Listing Users

```
ipmitool user list [<channel-number>]
```

Example:

```
ipmitool user list 1
```

Creating User

```
ipmitool user set name <user-id> <username>
```

For example:

```
ipmitool user set name 2 Admin
```

Setting User Password

```
ipmitool user set password <user-id> <password>
```

Example:

```
ipmitool user set password 2 AdminPass_123
```

Enabling/Disabling User

```
ipmitool user <enable|disable> <user-id>
```

Example:

```
ipmitool user enable 2
```

Setting User Privilege

```
ipmitool user priv <user-id> <privilege level(1-4)> [<channel-  
number>]
```

Where "privilege level":

- 1 – callback level (currently not supported)
- 2 – user level
- 3 – operator level
- 4 – administrator level

Example:

```
ipmitool user priv 2 0x3 1
```

Enabling Remote IPMI for User

To enable remote IPMI command functionality for a user:

```
ipmitool channel setaccess [<channel-number>] <user-id> ipmi=  
<on|off>
```

For example:

```
ipmitool channel setaccess 1 2 ipmi=on
```

Lanplus Commands to Execute IPMI Commands Remotely for Admin Users

Lanplus commands to execute IPMI commands remotely for users with admin permissions:

```
ipmitool -C 17 -I lanplus -U <user> -P <password> -H <bmc-ip-address> <ipmi-command>
```

For example:

```
ipmitool -C 17 -I lanplus -U ADMIN -P AdminPass_123! -H 10.10.10.10 user list 1
```

Lanplus Commands to Execute IPMI Commands Remotely for Non-admin Users

Lanplus commands to execute IPMI commands remotely for users with a non-administrator role:

```
ipmitool -C 17 -I lanplus -U <user> -P <password> -H <bmc-ip-address> -L <privilege (operator|user)> <ipmi-command>
```

For example:

```
ipmitool -C 17 -I lanplus -U operator1 -P operator123 -H  
10.10.10.10 -L operator user list 1  
ipmitool -C 17 -I lanplus -U user1 -P user123 -H 10.10.10.10 -L  
user chassis status
```

Deleting User

```
ipmitool user set name <user-id> ""
```

For example:

```
ipmitool user set name 2 ""
```

LLDP in Redfish

The Redfish chassis schema provides a structured and standardized way to represent essential information about the physical infrastructure of computing systems (the NVIDIA® BlueField® Platform for our purposes), offering valuable insights for system administrators, data center operators, and management software developers.

The LLDP schema provides the ability to enable/disable the LLDP in BMC and to get LLDP information from the BMC and from peer devices.

Getting LLDP Information

```
curl -k -u root:'<password>' -H 'Content-Type: application/json'
-X GET
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/DedicatedNetwork
```

Output example:

```

{
  "@odata.id":
  "/redfish/v1/Managers/Bluefield_BMC/DedicatedNetworkPorts/1",
  "@odata.type": "#Port.v1_7_0.Port",
  "Ethernet": {
    "LLDPEnabled": true,
    "LLDPReceive": {
      "ChassisId": "MAC: 8c:85:c1:a2:ae:80",
      "ChassisIdSubtype": 4,
      "ManagementAddressIPv4": "10.60.7.238",
      "ManagementVlanId": 95,
      "PortId": "Ifname: 1/1/5",
      "PortIdSubtype": 5,
      "SystemCapabilities": [
        "Bridge",
        "Router"
      ],
      "SystemDescription": "Aruba JL676A PL.10.13.0005",
      "SystemName": "MTL-T-F0-LAB-ORMANCE-SW-7-238"
    },
    "LLDPTransmit": {
      "ChassisId": "MAC: 94:6d:ae:5c:9d:cd",
      "ChassisIdSubtype": 4,
      "ManagementVlanId": 4095,
      "PortId": "MAC: 94:6d:ae:5c:9d:cd",
      "PortIdSubtype": 3,
      "SystemCapabilities": [
        "Station"
      ],
      "SystemDescription": "Linux dpu-bmc 5.15.50-a838e3d
#1 SMP Wed Mar 27 10:44:16 UTC 2024 armv7l",
      "SystemName": "dpu-bmc"
    }
  },
  "Id": "1",

```

```
"Links": {
  "EthernetInterfaces": [
    {
      "@odata.id":
"/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0"
    }
  ]
},
"Name": "Manager Dedicated Network Port"
}
```

Enabling/Disabling LLDP on BMC

```
curl -k -u root:'<password>' -H 'Content-Type: application/json'
-X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/DedicatedNetwork
-d '{"LLDPEnabled":<true/false>'}
```

Note

The IPMI stack in OpenBMC tries to workaround this IPMI spec implementation by creating a virtual interface for the VLAN ID specified by the user and then restricting IPMI to only access the newly created virtual interface. However, this solution has side effects like the LLDP tool being unable to obtain the VLAN interface ID because the LLDP tool works only with physical interfaces.

BMC Monitoring

This section is composed of the following subpages:

- [BMC FRUs](#)
- [Product Instance Identifier](#)
- [Software Versioning](#)
- [Storage Health Monitoring](#)

BMC FRUs

During the initialization process, the BMC scans all supported interfaces to locate available FRU devices. Each FRU device identified by the BMC is parsed and subsequently added to the BMC's list of FRU devices.

Info

Currently, the BMC is expected to detect the system FRU located at ID 0, as well as the BMC FRU.

The BMC FRU is available in several different revisions, each specific to the board based on its manufacturing date and type. These revisions are mainly identified by the unique product name given to each board.

FRU Field	Rev-1	Rev-2	Rev-3
FRU device description	Nvidia-BMCMezz (ID 169)	BlueField-3 DPU (ID 243)	BlueField SuperNIC (ID TBD)
Board manufacturing date	<Board-mfg-date>	<Board-mfg-date>	<Board-mfg-date>

FRU Field	Rev-1	Rev-2	Rev-3
Board manufacturer	Nvidia	Nvidia	Nvidia
Board product	Nvidia-BMCMezz	BlueField-3 DPU	BlueField SuperNIC
Board serial	<Board-serial>	<Board-serial>	<Board-serial>
Board part number	<Board-part-number>	<Board-part-number>	<Board-part-number>

BMC FRU Reading IPMI Commands

To retrieve FRU info, run:

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN fru print
```

Example output:

```
FRU Device Description : Nvidia-BMCMezz (ID 169)
Board Mfg Date         : Wed Nov  8 01:41:00 2023 UTC
Board Mfg              : Nvidia
Board Product          : Nvidia-BMCMezz
Board Serial           : MT2345300006
Board Part Number      : 900-9D3B6-00CV-AAA
Board Area Checksum    : OK
```

To print a specific FRU:

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN fru print
<fru_id>
```

FRU ID of the BMC FRU EEPROM is optional and can be found using the `fru print` command.

Product Instance Identifier

It is possible to set a unique name for the BMC. This name may be retrieve from Redfish without a password.

Setting Product Identifier

```
curl -k -u root:'<password>' -H 'Content-Type: application/json'
-X PATCH https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC -d
'{"ServiceIdentification": "<system_name>"}'
```

Retrieving Product Identifier

- To get the system name without a password:

```
curl -k -X GET https://<bmc_ip>/redfish/v1 | jq
'.ServiceIdentification'
```

- To get the system name with a password:

```
curl -k -u root:'<password>' -H 'Content-Type:
application/json' -X GET
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC | jq
'.ServiceIdentification'
```

Software Versioning

There is a software version for each of the BMC software components. You may retrieve this information by running the following for each component:

- Linux version – `uname -a` command from the Linux prompt
- OpenBMC version – `cat /etc/os-release` from the Linux prompt

Retrieving BMC Version Using Redfish

```
curl -k -u root:'<password>' -H 'Content-Type: application/json'
-X GET
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/BMC_Fi
{
  "@odata.id":
  "/redfish/v1/UpdateService/FirmwareInventory/BMC_Firmware",
  "@odata.type": "#SoftwareInventory.v1_4_0.SoftwareInventory",
  "Description": "BMC image",
  "Id": "BMC_Firmware",
  "Name": "Software Inventory",
  "RelatedItem": [],
  "RelatedItem@odata.count": 0,
  "SoftwareId": "",
  "Status": {
    "Conditions": [],
    "Health": "OK",
    "HealthRollup": "OK",
    "State": "Enabled"
  },
  "Updateable": true,
  "Version": "BF-23.09-1",
  "WriteProtected": false
}
```

Retrieving BMC Version Using IPMI

```
# ipmitool mc info
Device ID                : 1
Device Revision          : 1
Firmware Revision        : 23.09
IPMI Version             : 2.0
Manufacturer ID          : 33049
Manufacturer Name        : NVIDIA
Product ID               : 4 (0x0004)
Product Name             : Bluefield3 BMC
Device Available         : yes
Provides Device SDRs    : yes
Additional Device Support :
    Sensor Device
    SDR Repository Device
    SEL Device
    FRU Inventory Device
    IPMB Event Receiver
    Chassis Device
Aux Firmware Rev Info    :
    0x10
    0x01
    0x00
    0x00
```

Where the BMC version is formatted as `[Firmware Revision]-[Aux Firmware Rev Info 2nd byte]` which is 23.09-1 according to this example.

Storage Health Monitoring

The BMC continuously monitors system resources including CPU utilization, memory usage, and storage space. When these metrics exceed configured thresholds, warning or critical event logs are automatically generated to alert administrators.

Event logs can be viewed via the Redfish API at `/redfish/v1/Systems/system/LogServices/EventLog/Entries`.

CPU Metrics

The following table defines the thresholds for CPU utilization alerts:

Metric	Alert Type	Warning	Critical	Action	Description
CPU	Upper	>80%	>95%	Log only	Total BMC CPU utilization
CPU user	Upper	>80%	>95%	Log only	CPU time in user-space applications
CPU kernel	Upper	>80%	>95%	Log only	CPU time in kernel operations

Metric Log Example

To check the current event log for CPU alerts:

```
curl -k -u <usr>:<password> -H 'content-type: application/json' -X GET https://<bmc_ip>/redfish/v1/Systems/Bluefield/LogServices/EventLog,
```

Example output

```

{
  "@odata.id":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/468",
  "@odata.type": "#LogEntry.v1_15_0.LogEntry",
  "AdditionalDataURI":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/468/at
  "Created": "2026-01-19T15:57:22+00:00",
  "EntryType": "Event",
  "Id": "468",
  "Message": "CPU sensor crossed a critical high threshold
going high. Reading=96.881238 Threshold=95.000000.",
  "MessageArgs": [
    "CPU",
    "96.881238",
    "95.000000"
  ],
  "MessageId":
  "OpenBMC.0.4.SensorThresholdCriticalHighGoingHigh",
  "Name": "System Event Log Entry",
  "Resolution": "None",
  "Resolved": false,
  "Severity": "Critical"
}

```

Memory Metrics

Metric	Alert Type	Warning	Critical	Action	Description
Memory available	Lower	<30%	<10%	Log only	Memory available for applications
Memory shared	Upper	-	>35%	Log only	Shared memory usage

Storage Metrics

The BMC system provides real-time monitoring of read-write (RW) flash usage. You can query free storage space, receive notifications when usage crosses defined thresholds, and rely on automatic cleanup when limits are exceeded.

The following table defines the thresholds for storage usage alerts. Note that percentages refer to used space:

Metric	Alert Type	Warning	Critical	Path	Action	Description
Storage RW	Lower	<10%	<5%	<code>/run/initramfs/rw</code>	Auto cleanup	Root overlay filesystem; primary writable storage for BMC runtime data and configuration changes
Storage TMP	Lower	<20%	<5%	<code>/tmp</code>	Log only	Temporary files storage; used by services for transient data, cleared on reboot
Storage LOGGING	Lower	<30%	<20%	<code>/var/lib/logging</code>	Log only	Event logs and dump storage; contains Redfish logs, SEL entries, and debug dumps

Retrieving Free Storage Space

To check the current free RW flash space:

```
curl -k -H "X-Auth-Token: $token" -X GET  
https://${bmc}/redfish/v1/Managers/Bluefield_BMC/ManagerDiagnostic
```

Example output:

```
{
  "@odata.id" :
  "/redfish/v1/Managers/Bluefield_BMC/ManagerDiagnosticData",
  "@odata.type" :
  "#ManagerDiagnosticData.v1_2_0.ManagerDiagnosticData",
  "FreeStorageSpaceKiB" : 1488,
  "Id" : "ManagerDiagnosticData",
  "MemoryStatistics" : {
    "AvailableBytes" : 725983232,
    "BuffersAndCacheBytes" : 170594304,
    "FreeBytes" : 605347840,
    "SharedBytes" : 60747776,
    "TotalBytes" : 917188608
  },
  "Name" : "Manager Diagnostic Data",
  "ProcessorStatistics" : {
    "KernelPercent" : 0.6058,
    "UserPercent" : 0.5048
  },
  "ServiceRootUptimeSeconds" : 1282378.351
}
```

Storage Cleanup Notifications

When RW flash usage exceeds 90%, a Redfish event log entry is generated to alert that manual cleanup is required.

Example log:

```
{
  "@odata.id":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/7",
  "@odata.type": "#LogEntry.v1_15_0.LogEntry",
  "Created": "2025-09-15T13:30:43+00:00",
  "EntryType": "Event",
  "Id": "7",
  "Message": "Processes consuming HIGH Resource Storage_RW are
91%",
  "MessageArgs": [
    "Storage_RW",
    "91%"
  ],
  "MessageId": "OpenBMC.0.4.BMCSysResourceInfo",
  "Name": "System Event Log Entry",
  "Resolution": "None.",
  "Resolved": false,
  "Severity": "OK"
}
```

Automatic Cleanup

When RW flash usage exceeds 95%, the BMC automatically purges space by deleting:

- All dump files
- All event logs
- Files in home directories
- Files in system log directories

Warning

Exceeding 99% RW flash usage can make BMC functionality unstable and impede automatic cleanup.

After automatic cleanup, a Redfish event log entry is generated. For example:

```
{
  "@odata.id" :
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3",
  "@odata.type" : "#LogEntry.v1_15_0.LogEntry",
  "Created" : "2025-09-24T10:40:34+00:00",
  "EntryType" : "Event",
  "Id" : "3",
  "Message" : "RWFS cleanup completed.",
  "Modified" : "2025-09-22T10:40:34+00:00",
  "Name" : "System Event Log Entry",
  "Resolved" : false,
  "Severity" : "OK"
}
```

BMC Reset Control

This section is comprised of the following subpages:

- [Factory Reset BMC](#)
- [Reset or Reboot BMC](#)

Factory Reset BMC

The following commands factory reset the BMC configuration.

Factory Reset Redfish Command

```
curl -k -u root:"<PASSWORD>" -H "Content-Type: application/json"
-X POST
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Actions/Manager.F
-d '{"ResetToDefaultsType": "ResetAll"}'
```

Warning

Before connecting to the internet, it is important to change the default global password to prevent potential malicious attackers from hacking your system. For information on password policy, refer to section "[BMC Management Interface](#)".

i Info

After running factory reset, the BIOS configuration attributes list is updated only after rebooting the BlueField as the list gets its values from UEFI as BlueField is booting and Redfish is enabled.

Factory Reset IPMI Command

```
ipmitool raw 0x32 0x66
```

After issuing the `ipmitool raw` command for factory reset, you must log into the BMC and reboot it for the factory reset to take effect.

i Note

If you have lost your BMC login credentials and cannot login, you may issue the following command from the NVIDIA® BlueField® Arm:

```
ipmitool mc reset cold
```

In factory reset the time is also restarting, and time sync to the real time start after some of the services start. Because of that some of the services will show that there active time to be start at 1970.

Warning

Before connecting to the internet, it is important to change the default global password to prevent potential malicious attackers from hacking your system. For information on password policy, refer to section "[BMC Management Interface](#)".

Reset or Reboot BMC

Rebooting BMC Redfish Command

```
curl -k -u root:'<password>' -H "Content-Type: application/json"  
-X POST -d '{"ResetType": "GracefulRestart"}'  
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Actions/Manager
```

Rebooting BMC IPMI Command

```
ipmitool mc re cold
```

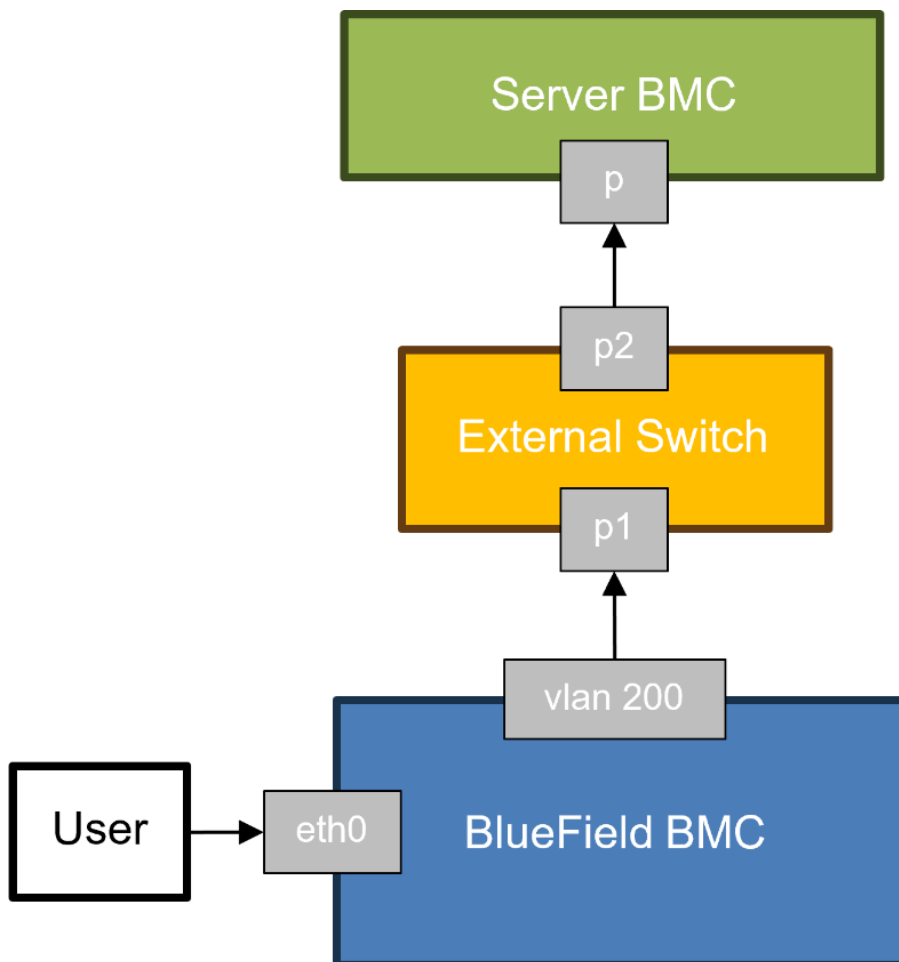
BMC Networking

This section is comprised of the following subpages:

- [Guest Tunnel](#)
- [Network Protocol Support](#)

Guest Tunnel

The NVIDIA® BlueField® BMC is capable of establishing a designated VLAN interface to forward IPMI or HTTPS traffic from an external source to a specific IP address within that VLAN. Users can set a specific IP address for their server BMC, which allows for the management of their server BMC via the BlueField BMC.



To enable this feature, users must configure their network according to the following:

- Assign the remote server's BMC IP address as 192.168.1.10 to enable traffic forwarding
- The VLAN ID for the guest tunnel is 200, thus the external switch linked to the BlueField RJ45 port (OOB) must be set up to accept packets tagged with VLAN 200

The BlueField BMC currently supports the following ports, which act as source ports to accept messages sent by users and forward them to the server BMC within the guest tunnel:

- 8443 – Port on BlueField BMC for accepting HTTPS messages
- 8623 – Port on BlueField BMC for accepting IPMI messages

Info

The guest tunnel is intended solely for debugging purposes. Users should refrain from sending large amounts of network traffic through the guest tunnel, as it may impact the performance of the BlueField BMC.

Querying Guest Tunnel Status

netfunc	cmd	data	Notes
0x3E	0xFD	0x0	Displays the current configuration: <ul style="list-style-type: none">• 0x01 – Disabled• 0x02 – Enabled

Disabling Guest Tunnel

netfunc	cmd	data	Notes
0x3E	0xFD	0x1	On success, returns: <ul style="list-style-type: none"> • 0x01 – Guest Tunnel is set to Disabled

Enabling Guest Tunnel

netfunc	cmd	data	Notes
0x3E	0xFD	0x2	On success, returns: <ul style="list-style-type: none"> • 0x02 – Guest Tunnel is set to Enabled

Example for enabling the guest tunnel on BlueField BMC:

```
#bmc> ipmitool raw 0x3e 0xfd 0x2
```

Example for sending the Redfish command to the guest tunnel:

```
#localhost> curl -k -u <bluefield_bmc_username> :
<bluefield_bmc_password> -H 'content-type: application/json' -X GET
https://<bluefield_bmc_ip>:8443/redfish/v1/Systems/Bluefield
```

Example for sending the IPMI command to the guest tunnel:

```
#localhost> ipmitool -p 8623 -C 17 -I lanplus -H
<bluefield_bmc_ip> -U <bluefield_bmc_username> -P
<bluefield_bmc_password> mc info
```

After receiving the responses from the Redfish and IPMI commands, the content of these responses originates from the server BMC, not the BlueField BMC.

Network Protocol Support

Note

To obtain the BMC MAC address, refer to the board label affixed to the NVIDIA® BlueField® device.

BMC management network interface can be configured using Redfish or IPMI. By default, BMC comes up with the DHCP network configuration.

Network configuration functions:

- Setting DHCP/Static network mode configuration
- Adding/setting IPv4/IPv6 configuration including IP address, gateway, netmask
- Adding DNS servers
- Adding NTP server
- Setting BMC time with NTP server or system RTC

Network Management Redfish Commands

Getting Network Protocol Configuration

```
curl -k -u root:'<password>' -X GET  
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol
```

Getting Interface Configuration

```
curl -k -u root:'<password>' -XGET
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfac
```

Enabling/Disabling Interface

```
curl -k -u root:'<password>' -XPATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfac
-d '{"InterfaceEnabled": <state>'
```

Where `<state>` can be `true` or `false`.

Note

Disabling the `eth0` interface on the BlueField BMC prevents OOB network functionality on the BMC. This inhibits the ability to execute any Redfish or IPMI commands through the network.

Configuring Static IPv4 Address

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfac
-d '{"IPv4StaticAddresses": [{"Address": "
<ip_addr>", "SubnetMask": "<netmask>", "Gateway": "<gw_ip_addr>"}]}'
```

Example:

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterface
-d '{"IPv4StaticAddresses": [{"Address": "10.7.7.7", "SubnetMask":
"255.255.0.0", "Gateway": "10.7.0.1"}]}'
```

Deleting Static IPv4 Address

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterface
-d '{"IPv4StaticAddresses": [null]}'
```

Enabling/Disabling IPv4 DHCP

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterface
-d '{"DHCPv4": {"DHCPEnabled": <state>}}'
```

Where `<state>` can be `true` or `false`.

Configuring Static DNS Server IPv4 and IPv6

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterface
-d '{"StaticNameServers": ["<dns_ip>"]}'
```

Configuring Static IPv6 Address

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfac
-d '{"IPv6StaticAddresses": [{"Address": "<ip>", "PrefixLength":
<len>}]}
```

Example:

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfac
-d '{"IPv6StaticAddresses": [{"Address":
"fe80::3eec:efff:fe3b:e02f", "PrefixLength": 64}]}'
```

Enabling/Disabling IPv6 DHCP

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfac
-d '{"DHCPv6": {"OperatingMode": "<state>"}}'
```

Where `<state>` can be:

- `Enabled` – DHCPv6 is enabled for this interface
- `Disabled` – DHCPv6 is disabled for this interface

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterface
-d '{"StatelessAddressAutoConfig": {"IPv6AutoConfigEnabled": "
<state>"}}'
```

Where `<state>` can be:

- `true` – Indicate IPv6 stateless address autoconfiguration (SLAAC) is enabled for this interface
- `false` – Indicate IPv6 stateless address autoconfiguration is disabled for this interface

Enabling/Disabling NTP

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol
-d '{"NTP": {"ProtocolEnabled": <state>}}'
```

Where `<state>` can be `true` or `false`.

Configuring Static NTP Server IP

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol
-d '{"NTP": {"NTPServers": ["<ntp_server_ip>"]}}'
```

Network Management IPMI Commands

The following subsections list the available network IPMI commands.

Configuring IPv4 Mode

The following command sets LAN channel 1 IP config mode to static or DHCP which corresponds to network interface `eth0`.

```
ipmitool lan set 1 ipsrc <mode>
```

Where `<mode>` can be `static` or `dhcp`.

Configuring IPv6 Mode

The following command sets LAN channel 1 IP config mode to static or DHCP which corresponds to network interface `eth0`.

```
ipmitool lan6 set 1 rtr_cfg <mode>
```

Where `<mode>` can be `static` or `dynamic`. `both` is unsupported.

Adding IPv4 Address

The following commands add IPv4 address, default gateway, and netmask to the network interface `eth0`.

- IP address:

```
ipmitool lan set 1 ipaddr <ip-address>
```

- Default gateway:

```
ipmitool lan set 1 defgw ipaddr <ip-address>
```

- Netmask:

```
ipmitool lan set 1 netmask <netmask>
```

Note

IPMI supports only a single static IP address. If multiple static IP addresses are configured on the system, the new netmask will be applied to only one of them.

Getting IPv4 Config

The following command gets IPv4 network config for channel 1 which corresponds to the network interface `eth0`.

```
ipmitool lan print 1
```

Setting IPv6 Address

The following command adds IPv6 address to the network interface `eth0`.

```
ipmitool lan6 set 1 nolock static_addr 0 enable <ipv6-address> 64
```

Getting IPv6 Config

The following command gets IPv6 network config for channel 1 which corresponds to the network interface `eth0`.

```
ipmitool lan6 print 1
```

Getting DNS Server

```
ipmitool raw 0x32 0x6B
```

Output:

```
0b 31 30 2e 31 35 2e 31 32 2e 36 37
```

This output corresponds to `10.15.12.67`.

Adding DNS Server

```
ipmitool raw 0x32 0x6C 0x0b 0x31 0x30 0x2e 0x31 0x35 0x2e 0x31
0x32 0x2e 0x36 0x37
```

Output:

```
0x0b 0x31 0x30 0x2e 0x31 0x35 0x2e 0x31 0x32 0x2e 0x36 0x37
```

This output corresponds to `10.15.12.67`.

Getting NTP Server

```
ipmitool raw 0x32 0xA7
```

Output:

```
01 11 31 2e 69 6e 2e 70 6f 6f 6c 2e 6e 74 70 2e 6f 72 67
```

Where:

- `01` – NTP status enable/disable
- `11` – NTP server length
- `31 2e 69 6e 2e 70 6f 6f 6c 2e 6e 74 70 2e 6f 72 67` – NTP server address byte stream which corresponds to `1.in.pool.ntp.org`

Adding NTP Server

```
ipmitool raw 0x32 0xA8 0x01 0x31 0x2e 0x69 0x6e 0x2e 0x70 0x6f  
0x6f 0x6c 0x2e 0x6e 0x74 0x70 0x2e 0x6f 0x72 0x67
```

Where:

- `31 2e 69 6e 2e 70 6f 6f 6c 2e 6e 74 70 2e 6f 72 67` – NTP server address byte stream which corresponds to `1.in.pool.ntp.org`

Enabling NTP Time Sync

The following command enables time sync to NTP server.

```
ipmitool raw 0x32 0xA8 0x02 0x01
```

Where:

- `0x01` – enable NTP

Disabling NTP Time Sync

The following command disables time sync to NTP server.

```
ipmitool raw 0x32 0xA8 0x02 0x00
```

Where:

- `0x00` – disable NTP

Configuring Router IPv6 Mode

The following command sets router mode to static or DHCP.

```
ipmitool lan6 set 1 rtr_cfg <mode>
```

Where `<mode>` can be:

- static
- Dynamic

Note

Configuring static mode also requires setting the static router IP and static router MAC address.

Note

Router prefix can only be 0.

Configuring IPv6 Static Router IP

The following command sets the IPv6 address for the static router.

```
ipmitool raw 0x0c 0x01 0x01 0x41 <ip-hex>
```

Where:

- `<ip-hex>` – the IP address

Configuring IPv6 Static Router MAC

The following command sets the MAC address for the static router.

```
ipmitool raw 0x0c 0x01 0x01 0x42 <mac-hex>
```

Where:

- `<mac-hex>` – the IP MAC address

BMC Redfish

This section is comprised of the following subpages:

- [BlueField BMC Redfish Triggers](#)
- [Redfish Certificate Management](#)

BlueField BMC Redfish Triggers

Redfish triggers allow the user to get a journal message when a certain metric crosses a defined threshold for a defined time:

- The trigger threshold can only be a numeric threshold
- The trigger thresholds are unrelated to the sensor thresholds
- The maximum number of triggers allowed in the system is 10

For more details, refer to [Redfish Resource and Schema Guide](#).

Adding Numeric Trigger

Adds a numeric trigger to the BMC.

```
curl -k -u root:'<password>' -H "Content-Type: application/json"
-X POST https://<bmc_ip>/redfish/v1/TelemetryService/Triggers/ -d
'{"Id": "< >", "Name": "<>", "MetricType": "<>", "TriggerActions": [
<> ], "NumericThresholds": { "<>": { "Activation": "<>", "DwellTime":
<>", "Reading": "<>"} }, "MetricProperties": [ "<>" ]}'
```

Deleting Numeric Trigger

```
curl -k -u root:'<password>' -H "Content-Type: application/json"
-X DELETE
https://<bmc_ip>/redfish/v1/TelemetryService/Triggers/<trigger-
name>
```

Redfish Certificate Management

Certificate management actions—such as retrieving certificate information or performing atomic certificate replacement—are accessible through the `CertificateService` resource.

The `CertificateLocations` resource provides an inventory of all certificates managed by the service.

For additional details, refer to the [Redfish Certificate Management White Paper](#).

Common Certificate Management Commands

Getting Certificate Locations

Inventory of all certificates the service is managing.

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/CertificateService/CertificateLocations
```

Root CA Management Commands

List Root CA

```
curl -k -u root:'<password>' -X GET  
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Truststore/Cert:
```

Getting Certificate Information

```
curl -k -u root:'<password>' -X GET  
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Truststore/Cert:
```

Installing Root CA Certificate

```
curl -k -u root:'<password>' -X POST  
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Truststore/Cert:  
-d @rootca.json
```

Replacing Existing Root CA Certificate

```
curl -k -u root:'<password>' -X PATCH  
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Truststore/Cert:  
-d @rootca.json
```

Root CA Certificate Creation and Replacement

1. Generate Root CA certificate:

```

cat > root-ca.cnf << EOF
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
prompt = no

[req_distinguished_name]
C = <country>
ST = <state>
L = <location>
O = OpenBMC
OU = bmcweb
CN = <common_name>

[v3_req]
basicConstraints = critical,CA:true
keyUsage = critical,keyCertSign,cRLSign
subjectKeyIdentifier = hash
EOF

# Generate root CA key
openssl genrsa -out root-ca-key.pem <key_size>

# Generate root CA certificate
openssl req -x509 -new -nodes \
    -key root-ca-key.pem \
    -sha256 -days <validity_days> \
    -out root-ca-cert.pem \
    -config root-ca.cnf \
    -extensions v3_req

```

2. Create a JSON file for the root CA certificate add

```
{
  "CertificateString": "<cert_string>",
  "CertificateType": "PEM"
}
```

3. Install the root CA certificate (can have more than 1).

```
curl -k -u root:'<password>' -X POST
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Truststore/Certificates -d @rootca.json
```

Server Certificate Management Commands

Getting Certificate Information

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol
```

Replacing Existing Certificate

```
curl -k -u root:'<password>' -X POST
https://<bmc_ip>/redfish/v1/CertificateService/Actions/CertificateS
-d @certificate.json
```

Generating CSR

Generate certificate signing request (CSR):

```
curl -k -u root:'<password>' -H "Content-Type: application/json"
-X POST
https://<bmc_ip>/redfish/v1/CertificateService/Actions/CertificateS
-d @csr_file.json
```

Installing Certificate

```
curl -k -u root:'<password>' -H "Content-Type: application/octet-
stream" -X POST
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol
-d @certificate.json
```

Example for CSR Generation, Certificate Creation and Replacement

1. Configure your CA to include at least the following extensions for the signed TLS server certificates:

```
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = IP:192.168.240.1
```

Note

The extension `subjectAltName = IP:192.168.240.1` is mandatory.

2. Create a JSON containing the subject data for the BlueField BMC to use when creating the CSR. For example:

```
{
  "City": "<city>",
  "CertificateCollection": {
    "@odata.id":
"/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol/HTTPS/Cert
  },
  "CommonName": "bmc0123456789.mycompany.com",
  "Country": "<country>",
  "Organization": "<company_name>",
  "OrganizationalUnit": "<my_org>",
  "State": "<state>",
  "KeyPairAlgorithm": "EC"
}
```

3. Generate a certificate signing request using the `forth` command in the table above and the JSON file created in the previous step:

Info

The BMC replies with a JSON containing the CSR.

```

curl -k -u root:'<password>' -H "Content-Type:
application/json" -X POST
https://<bmc_ip>/redfish/v1/CertificateService/Actions/Certifi
-d @csr_file.json
{
  "CSRString": "-----BEGIN CERTIFICATE REQUEST-----\
<CSR_DATA>\n-----END CERTIFICATE REQUEST-----\n",
  "CertificateCollection": {
    "@odata.id":
"/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol/HTTPS/Cert
  }
}

```

4. Extract the CSR string from the JSON and sign the CSR using your CA. For example, this is how to include the required extensions to the signed TLS server certificates:

```

openssl x509 -req -in bmc.csr -CA CA-cert.pem -CAkey CA-
key.pem -CAcreateserial -out bmc.crt -days 3650 -sha384 -
extfile extfile.txt

```

Where:

- `bmc.csr` contains the CSR string from the previous step
- `CA-cert.pem` contains the CA certificate to be used to sign the CSR
- `CA-key.pem` contains the CA private key
- `extfile.txt` contains the extensions mentioned in the first step (`basicConstraints`, `keyUsage`, and `subjectAltName`)
- `bmc.crt` is the output file which will contain the BMC certificate signed by the CA

5. Create a JSON file for the BlueField BMC signed TLS server certificate data:

```
{
  "CertificateString": "-----BEGIN CERTIFICATE-----
\n<bmc.crt-data>\n-----END CERTIFICATE-----",
  "CertificateType": "PEM",
  "CertificateUri":
  {
    "@odata.id":
    "/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol/HTTPS/Cert
  }
}
```

6. Replace the BMC certificate using the [third](#) command in the table above and the JSON created in the previous step.

```
curl -k -u root:'<password>' -X POST
https://<bmc_ip>/redfish/v1/CertificateService/Actions/Certifi
-d @certificate.j
```

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF

ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright 2026. PDF Generated on 02/28/2026